heig-vd

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD

www.heig-vd.ch

Obfuscator-LLVM — Software Protection for the Masses

19-05-2015

Hes·so
Haute Ecole Spécialisée
de Suisse occidentale

heig-vd

Team behind Obfuscator-LLVM

Joint work with

- Julien Rinaldini (HES-SO)
- Johan Wehrli (HES-SO)
- Julie Michielin (Nagravision)
- Pascal Junod (HES-SO)

and a few others bachelor and master students over the past 3 years.

heig-vd

Why software protection?

- Cryptography solves many information security problems… but surely not all!
- How to secure cryptographic SW implementations?
  - Side-channel resistance
  - Resistance to white-box adversaries
    - HW root of trust
    - SW-only techniques

heig-vd

Challenges behind Software Protection

- Source vs. binary protection
- Interpreted vs. native code
- Supported languages
- Supported target platforms

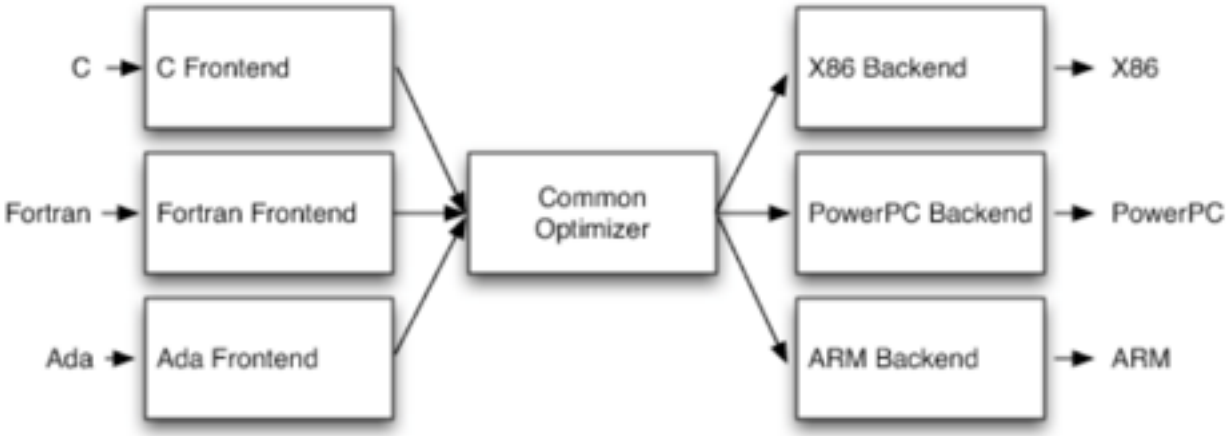
heig-vd

The LLVM Compilation Framework

- Open-source compilation suite
- Project initiated by Chris Lattner in 2006, then supported by many big companies
- Main competitor of the GNU Compiler Collection
- Supports C/C++, Objective-C, Fortran, Swift + some others
- Supports x86, PowerPC, ARM, Sparc, Alpha, MIPS + others

heig-vd

The LLVM Compilation Framework



heig-vd

Obfuscator-LLVM Goals

- Write an obfuscation tool that is independent of the programming language
- Be as much as possible independent of the target platform
- Not always possible:
  - Tamper-proofing
  - Anti-debugging tricks insertion
  - Anti-disassembly tricks insertion

heig-vd

Obfuscator-LLVM Current Functionalities

- Code diversification
- Instructions substitution
- Code flattening
- Tamper-proofing
- Procedure merging
- Bogus control-flow insertion
- (Constant scrambling)
- (Anti-debugging tricks insertion)

heig-vd

Code Diversification

- Compilation is inherently deterministic
- Use of a cryptographic PRNG (AES in CTR mode)
- Fixing the seed value makes the obfuscation process deterministic
- Use as much entropy as possible in the transformations
- Can be interpreted as a first step towards binary watermarking.

heig-vd

Instructions Substitution

- Simple instruction substitution, sometimes involving random values

Operator	Equivalent Instruction Sequence

`a = b + c`	`a = b - (-c)`
	`a = -(-b+ (-c))`
	`a = b + r; a += c; a -= r`
	`a = b - r; a += c; a += r`

`a = b - c`	`a = b + (-c)`
	`a = b + r; a -= c; a -= r`
	`a = b - r; a -= c; a += r`

`a = b & c`	`a = (b^ !c) & b`
`a = b | c`	`a = (b&c) | (b^ c)`
`a = b ^ c`	`a = (!b&c) | (b&!c)`

heig-vd

Instructions Substitution

- Can partly be interpreted as simple data masking
- Floating point arithmetic not supported (loss of numeric accuracy)
- Several iterations possible
- Fine-tuning of substitution density per basic-block

heig-vd

Code Flattening

- Several variants implemented
- Robustness depends on the (un-) predictability of the routing variable
- Can be combined with basic-bloc splitting to increase the complexity

heig-vd

Tamper-Proofing

- Code hashing based on on-line computation of CRCs
- GF(2)-linear nature of CRC allows interleaved checked
- Use of dedicated instructions on x86
- Result of CRC is injected in the code flattening routing variable
  - Tampered code results in an infinite loop
- Need a platform-dependent post-processing step

heig-vd

Procedure Merging

```
// Module test.c
int foo(int a) {
  return a+2;
}

float bar(float a) {
  return a+2.0;
}
```


heig-vd

Procedure Merging

```
// Module test.c
void merged(int sw, void *ret, ...) {
  switch(sw) {
    case 0:
            va_list ap;
            va_start(ap, 1);
            int a = va_args(ap, int);
            va_end(ap);
            int *b = (int*)ret;
            *b = a+2;
            break;
    case 1:
            va_list ap;
            va_start(ap, 1);
            float a = va_args(ap, float);
            va_end(ap);
            float *b = (float*)ret;
            *b = a+2.0;
            break;
  }
  return;
}
```


heig-vd

Function Merging

- Initial functions are replaced by wrappers
- Unique goal of a wrapper is to deal with the parameters, to feed the merged function and to recover the return value.
- Selection variable is a salted cryptographic hash of the initial symbol name

Bogus Control-Flow Insertion

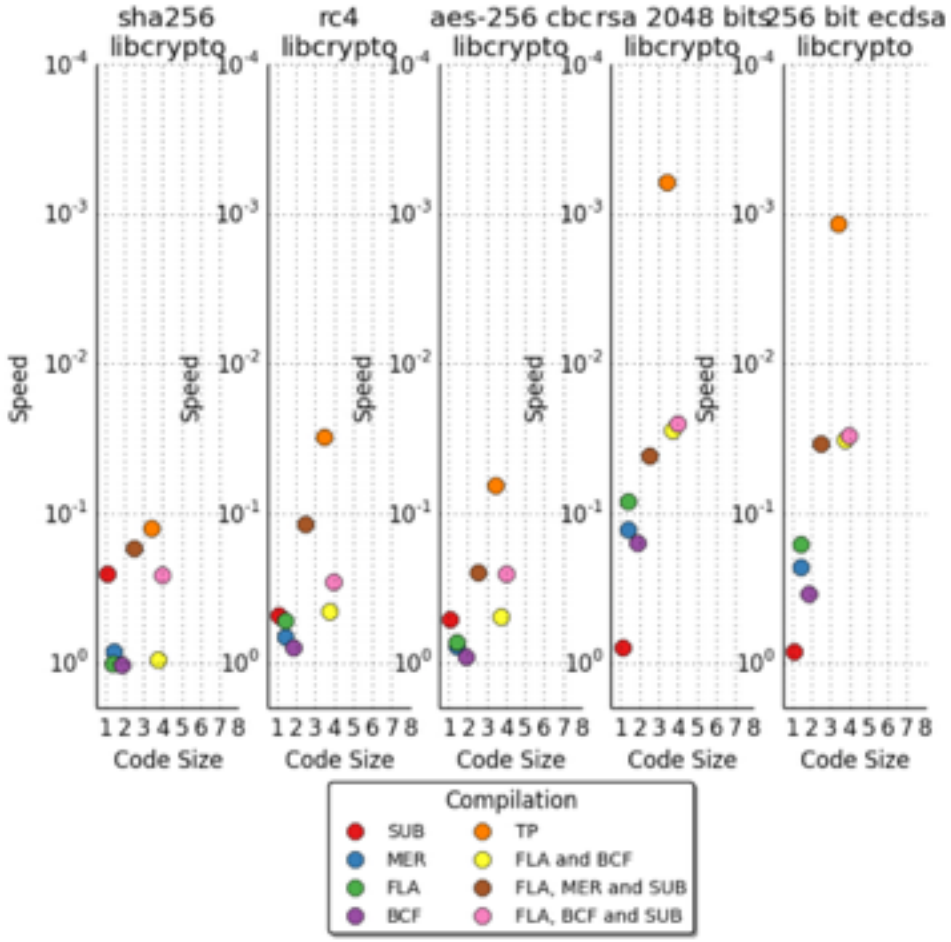
- Insertion of bogus control-flow constructs
- Use of opaque predicates
- Junk code is legit code slightly modified

Testing

- Test suites of
  - sqlite (~ 180’000 tests)
  - libtomcrypt
  - OpenSSL
- Debugging an obfuscator is … hard!

heig-vd

Performances



heig-vd

Open-Source or Not ?

About Obfuscator-LLVM, Dual-Use Tools and Academic Ethics

Dec 24 2013

129 Tweet 28 J'aime 4 +1

On November 25th, my team has announced the release for Christmas of Obfuscator-LLVM, an open-source obfuscation tool based on the LLVM compilation suite. Why Christmas? Because version 3.4 of LLVM was planned to be released on the 23rd of December, and we wished to port our code to the latest version of the compiler before publishing it.

On December 18th, I have been privately contacted by pod2g, a French security researcher active in the Apple jailbreaking scene, kindly asking me whether his team, the evad3rs, could get an early access to our tool. Without thinking too much about the possible consequences, and naively seeing it as an easy way to get some publicity for our research project, my collaborators and I have accepted to send them the source code of Obfuscator-LLVM one week earlier than the planned release. In exchange, we only asked to be credited on their website; I would like to clearly state that we never spoke about financial compensation, or about any other kind of reward.

The evasion jailbreak was released on December 23rd, taking the jailbreak community by surprise, and instantly generating a controversy. Indeed, the jailbreaking software arrived bundled with a Chinese app store apparently delivering pirated apps and/or malware.

Providing a version of Obfuscator-LLVM to the evad3rs one week in advance on our planned release was a mistake, and we regret this turn of events, as our academic research project is now somehow linked to the murkier side of ITsec.

heig-vd

Open-Source or not?

- The open-source version of Obfuscator-LLVM is a « light » one.
  - Substitutions
  - Bogus control-flow insertion
  - Control-flow flattening
- We still hope it will be useful for academic purposes
- Please contribute !

heig-vd

Q&A

- @cryptopathe
- @ollvm
- http://o-llvm.org